(h)

HDL INPUT METHOD AND DEVICE THEREFOR

Patent Number: JP2000132584
Publication date: 2000-05-12

Inventor(s): HARA NAOKI; MORIWAKI IKU
Applicant(s):: HITACHI LTD

Requested Patent: ☐ JP2000132584 (JP00132584)
Application Number: JP19980304994 19981027
Priority Number(s):
IPC Classification: G06F17/50
EC Classification:
Equivalents:

Abstract

PROBLEM TO BE SOLVED: To provide an HDL description input system reducing the burden on a designer for the HDL(hardware description language) description and preventing a grammar error of the designer.
SOLUTION: An HDL syntax data base 104a having the models of respective HDL syntaxes and an associated design data base 104b having connection information between circuits are installed. An HDL syntax generation part 100c generating the HDL syntax from the data base 104a with the reservation word of HDL which a designer inputs as a key, an associated description generation part 100d generating the HDL syntax from the model of the HDL syntax and circuit information when circuit information required for input exists in the data base 104b, an associated description correction part 100e correcting an affected HDL text file from the data base 104b when the correction of the generated HDL text file 104c affects the other HDL text file and a semantics check part 100f checking semantics having no problem in terms of a circuit, are installed.

(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開２０００－１３２５８４
（Ｐ２０００－１３２５８４Ａ）
(43)公開日　平成12年5月12日(2000.5.12)

(51)Int.Cl.⁷	識別記号	FI		テーマコード(参考)
Ｇ０６Ｆ　17/50		Ｇ０６Ｆ　15/60	６５４　Ｒ	5B046
			６６４　Ｚ	

審査請求　未請求　請求項の数５　ＯＬ　（全１２頁）

(21)出願番号　特願平10-304994

(22)出願日　平成10年10月27日(1998.10.27)

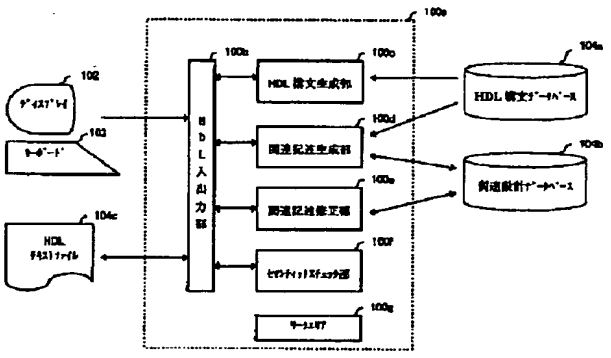
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目6番地
(72)発明者　原　直樹
神奈川県秦野市堀山下1番地　株式会社日立インフォメーションテクノロジー内
(72)発明者　森脇　郁
神奈川県川崎市幸区鹿島田890番地　株式会社日立製作所システム開発本部内
(74)代理人　100068504
弁理士　小川　勝男

Ｆターム(参考)　5B046 AA08 BA03 JA01 KA03 KA05

(54)【発明の名称】ＨＤＬ入力方法及び装置

(57)【要約】　(修正有)
【課題】設計者のＨＤＬ（ハードウェア記述言語）記述の負担の低減と文法ミスを防ぐＨＤＬ記述入力システムを提供する。
【解決手段】各ＨＤＬ構文の雛形を持ったＨＤＬ構文データベース１０４ａと、各回路間の接続情報を持った関連設計データベース１０４ｂを備え、設計者が入力したＨＤＬの予約語を鍵にデータベース１０４ａからＨＤＬ構文を生成するＨＤＬ構文生成部１００ｃと、データベース１０４ｂに入力に必要な回路情報がある場合にＨＤＬ構文の雛形と回路情報からＨＤＬ構文を生成する関連記述生成部１００ｄと、既作成済みのＨＤＬテキストファイル１０４ｃの修正が他のＨＤＬテキストファイルに影響する場合、データベース１０４ｂから影響するＨＤＬテキストファイルの修正を行う関連記述修正部１００ｅと、回路的に問題がないセマンティックスチェックを行うセマンティックスチェック部１００ｆを備える。

図２



1

【特許請求の範囲】

【請求項１】ハードウエア記述言語（以下HDLと言う）を用いて回路設計を行うためのＨＤＬ入力装置において、各HDLの構文の雛形を持ったデータベースと、各回路間の接続情報を持ったデータベースを参照する手段を備え、設計者が入力したHDLの予約語からＨＤＬ構文を生成するプログラムと、回路間の関連情報を持ったデータベースから回路情報を生成しHDL構文を生成するプログラムと、既存のHDLテキストファイルの変更が他のHDLテキストファイルに関連している場合関連個所を修正するプログラムと、回路のセマンティックスチェックを行うプログラムを備えたＨＤＬ入力装置。

【請求項２】前記請求項１のHDL入力装置において、設計者が入力したHDLの予約語を鍵に、HDL構文の雛形を持つデータベースを検索し、雛形から予約語のHDL構文を生成するプログラム。

【請求項３】前記請求項１のHDL入力装置において、回路間の接続情報を持つデータベースに入力に必要な回路情報が登録されている場合に、HDL構文の雛形と回路情報からHDL構文を生成するプログラム。

【請求項４】前記請求項１のHDL入力装置において、機能変更や論理不良等により既作成済みのHDLテキストファイルの変更を行うことで他のHDLテキストファイルも変更が必要ないか回路間の接続情報を持つデータベースを検索し変更が必要な場合他のHDLテキストファイルも変更を行うプログラム。

【請求項５】前記請求項１のHDL記述入力装置において、未使用な信号を記述や、機能レベル／ゲートレベルのシミュレーション結果に影響があるprocess文、always文のセンシティビテリストのセマンティックスチェックを行うプログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、HDLを用いた回路設計の開発に関するものである。

【０００２】

【従来の技術】半導体製造技術の進展により１チップ上に搭載できるゲート数は増加し続けている。こうした状況からこれまでのゲートレベルの回路図入力による設計手法に代わって、ゲートレベルより上位の機能レベルを表現可能なHDLによる設計手法に移行している。ＨＤＬ設計手法では高度な記述が可能な反面、設計者がＨＤＬ文法を学習し、シンタックスに注意しながら機能を満たす回路記述の入力を行わなければならないと言う繁雑さがある。

【０００３】上記の如き問題を解決する一手法として、日本特許特開平６－３１４３１４では回路の機能と生成される回路の目標値を与えてＨＤＬを作成する入力装置が示されている。

【０００４】

2

【発明が解決しようとする課題】HDLを用いた回路設計では、設計者がHDL文法を学習し入力を行わなければならない。

【０００５】そのためHDL文法のシンタックスに注意しながら入力をしなければならなく、HDL設計の経験が少ないと記述ミスを起こし易い。

【０００６】また論理不良や仕様変更による記述に変更が発生し、その変更が関連して他の記述個所も修正が必要な場合、HDL設計に熟練した設計者でも見逃す可能性がある。

【０００７】日本特許特開平６－３１４３１４では、記述の自由度が小さく、論理変更等による関連個所の記述の変更に対応していない。

【０００８】本発明は、前記の従来の問題点を解決し設計者の負担を減らしたＨＤＬ入力システムである。

【０００９】

【課題を解決するための手段】上記課題を解決するため、本発明では、各HDL構文の雛形を持ったデータベースと、各回路間の接続情報を持ったデータベースを参照する手段を備え、設計者が入力したHDLの予約語を鍵にHDL構文の雛形情報を持ったデータベースからH DL構文を生成する手段と、回路間の接続情報を持つデータベースに入力に必要な回路情報がある場合にHDL構文の雛形と回路情報からHDL構文を生成する手段と、既作成済みのHDLテキストファイルの修正が他のHDLテキストファイルに影響する場合、回路間の接続情報を持つデータベースから影響するHDLテキストファイルの修正を行う手段と、回路的に問題がないかセマンティックスチェックを行う手段を提供する。

【００１０】

【発明の実施の形態】以下、本発明の実施形態を図を用いて説明する。

【００１１】＜図１の説明＞図１は、HDL記述入力装置の構成図である。

【００１２】HDL入力プログラム１００ａは主記憶装置１００に格納される。主記憶装置１００に格納したHDL入力プログラム１００ａを処理するCPU１０１、ＨＤＬ入力プログラムの入力画面を表示するディスプレイ１０２、設計者がHDLの入力を行うためのキーボード１０３、ＨＤＬの予約語からＨＤＬ構文を生成するための雛形情報を持つHDL構文データベース１０４ａと、機能変更や論理不良等で既作成済みのHDLテキストファイル１０４ｃの修正により他のHDLテキストファイル１０４ｃも修正する必要がある場合、その他のHDLテキストファイル１０４ｃの修正個所を検出しＨＤＬを修正するための回路の入出力端子情報と関連回路名称情報を持つ関連設計データベース１０４ｂとHDL入力プログラム１００ａから生成したＨＤＬテキストファイル１０４ｃを保存するハードディスク１０４から構成され、それぞれの装置はバス１０５で接続されている。

【００１３】設計者（操作者）はキーボード１０３からＨＤＬの予約語や回路情報の入力を行う。ディスプレイ１０２は設計者が入力したＨＤＬの文字列（予約語や論理記述）がＨＤＬ入力プログラム１００ａの入力画面に表示される。ＨＤＬ入力プログラム１００ａはキーボード１０３から入力された文字列を入力データとしＣＰＵ１０１でＨＤＬ入力プログラム１００ａの命令が実行される。命令によりハードディスク１０４に格納されているＨＤＬ構文データベース１０４ａや関連設計データベース１０４ｂの情報をもとにＨＤＬ構文の生成を行う。

【００１４】設計者がＨＤＬ記述が完成したと判断したら設計者はＨＤＬ入力プログラム１００ａにＨＤＬテキストファイル１０４ｃの出力指示を行いハードディスク１０４に出力する。

【００１５】＜図２の説明＞図２は、HDL入力プログラム１００ａの構成図である。

【００１６】ＨＤＬ入出力部１００ｂはキーボード１０３から入力したHDLの予約語である文字列データを受け取りる。次に文字列データを各処理部に渡しHDL入力プログラム１００ａの制御を行う。またHDL入力プログラム１００ａの処理によって作成されたＨＤＬテキストファイル１０４ｃをハードディスク１０４に出力する。

【００１７】ＨＤＬ構文生成部１００ｃの入力データはＨＤＬ入出力部１００ｂから受け取った文字列データ（予約語）である。受け取った文字列データを鍵にＨＤＬ構文データベース１０４ａを参照し、該当する文字列データの雛形情報からＨＤＬ構文を生成しＨＤＬ入出力部１００ｂに返す。

【００１８】関連記述生成部１００ｄはＨＤＬ入出力部１００ｂから階層記述に関する文字列データ（予約語と回路名称）を受け取る。受け取った文字列データ（回路名称）を鍵に関連設計データベース１０４ｂを検索して文字列データ（回路名称）があれば関連設計データベース１０４ｂからＨＤＬ構文に必要な情報を作成し、また文字列データ（予約語）を鍵にＨＤＬ構文データベース１０４ａからＨＤＬ構文の雛形情報を抽出し関連設計データベース１０４ｂとマージしてＨＤＬ構文を生成しＨＤＬ入出力部１００ｂに返す。

【００１９】関連記述修正部１００ｅは既に作成済みのＨＤＬテキストファイル１０４ｃを修正する場合に処理を行う。既に作成したＨＤＬテキストファイル１０４ｃの入出力端子記述に対して端子名の変更や端子の追加または削除を行う場合に、ＨＤＬ入出力部１００ｂから回路名称と端子情報を受け取る。その端子が変更したＨＤＬテキストファイル１０４ｃ以外に記述されていないか端子を変更をした回路名称を鍵に関連設計データベース１０４ｂを検索する。検索の結果他のＨＤＬテキストファイル１０４ｃで使用していたら変更した端子と同じ様に記述の変更を行いＨＤＬ入出力部１００ｂに返す。

【００２０】セマンティックスチェック部１００ｆはＨＤＬ入出力部１００ｂから設計者が入力した信号宣言・入出力端子情報の文字列データを入力データとする。信号宣言・入出力端子情報と論理記述部の信号端子情報のそれぞれをHDL入力プログラム１００ａのワークエリア１００ｇに格納する。そしてそれぞれの信号・端子情報を比較し過不足な信号・端子の記述がないかチェックを行い過不足があった場合はメッセージの出力を行う。

【００２１】＜図３の説明＞図３は、HDL構文データベース１０４ａのデータ構造を示す。

【００２２】ＨＤＬ構文データベース１０４ａは、ＨＤＬ構文生成部１００ｃ・関連記述生成部１００ｄでＨＤＬ構文を生成するために使用する。ＨＤＬ構文データベース１０４ａは、設計者が入力したＨＤＬの文字列（予約語）を鍵にして、ＨＤＬ構文データベース１０４ａの予約語３０２の検索を行う。設計者が入力した文字列と一致する予約語３０２があれば、予約語に対応する構文情報（雛形）３０３からＨＤＬ構文を生成する。

【００２３】ＨＤＬ構文データベース１０４ａはＶＨＤＬの予約語かＶｅｒｉｌｏｇＨＤＬの予約語か識別するための言語種別３０１と、ＶＨＤＬまたはＶｅｒｉｌｏｇＨＤＬの予約語３０２と、予約語３０２の構文情報（雛形）３０３から構成される。

【００２４】例えば設計者が'entity'と入力したとする。ＨＤＬ構文生成部１００ｃは入力された文字列'entity'を鍵にＨＤＬ構文データベース１０４ａの言語種別３０１ＶＨＤＬ・予約語３０２'entity'の検索を行う。検索の結果'entity'が検出されたら'entity'構文情報（雛形）３０３からＨＤＬ構文を生成する。

【００２５】ＨＤＬ構文データベース１０４ａは予めデータベースとして準備しておく。

【００２６】＜図４の説明＞図４は、関連設計データベース１０４ｂのデータ構造を示す。

【００２７】関連設計データベース１０４ａは、関連記述生成部１００ｄと関連記述修正部１００ｅで使用される。関連設計データベース１０４ａは入力したＨＤＬ記述の回路名称４０１とその回路の端子名称４０２と端子方向４０３と端子属性４０４と関連回路情報４０５から構成される。回路名称４０１は回路に対してつけられた名称である。VHDLであれば予約語‘entity’で宣言した回路名称である。またｖｅｒｉｌｏｇＨＤＬの場合は予約語‘module’で宣言した回路名称である。

【００２８】端子名称４０２は回路に付けられた入出力端子の名称である。端子方向４０３は端子が入力端子（ＶＨＤＬの場合ｉｎ、ｖｅｒｉｌｏｇＨＤＬの場合ｉｎｐｕｔ等）・出力端子（ＶＨＤＬの場合ｏｕｔ、ｖｅｒｉｌｏｇＨＤＬの場合ｏｕｔｐｕｔ等）であるか判別するための端子方向の情報である。端子属性４０４は端子の属性を表わす。たとえばstd_logic型やstd_logic_vector型などの信号の型を表わす。端子名称４０２・端子方向４０３・端子属性４０４は回路の端子本数分生成

する。

【００２９】関連設計データベース１０４ｂの登録は、回路記述が終了してＨＤＬテキストファイル１０４ｃの出力時に回路名称４０１・端子名称４０２・端子方向４０３・端子属性４０４をＨＤＬテキストファイル１０４ｃから抽出し登録処理する。ＶＨＤＬの場合回路名称は'entity'文から、端子情報は'port'文から作成する。またｖｅｒｉｌｏｇＨＤＬの場合回路名称は'modul'文から、'input'、'output','inout'文から作成する。関連回路情報４０５は他の関連記述生成部１００ｄで既作成済みHDLテキスタファイル１０４ｃの回路端子を使用した場合に、参照しているＨＤＬテキストファイル１０４ｃの回路名称を登録する。

【００３０】図４の場合回路名'ADD'は端子名称４０２'ＩＮ１'・端子方向４０３'in'・端子属性４０４'std_logic'の端子を持ち、回路名称４０１'ADD'は回路名topが参照している事を示している。

【００３１】また関連記述修正部１００ｅで端子情報に変更があった場合に関連設計データベース１０４ｂの端子名称４０２・端子方向４０３・端子属性４０４も変更に合わせて更新される。

【００３２】＜図５の説明＞図５は、HDL構文生成部１００ｃの原理をあらわす。

【００３３】ＨＤＬによる回路設計を行う場合、設計者はＨＤＬの文法を学習しなければならなず、ＨＤＬ設計の経験の少ない設計者だとシンタックスミスをおかしやすいという問題がある。

【００３４】そこでHDL構文生成部１００ｃは設計者が入力したＨＤＬの予約語を鍵にＨＤＬ構文データベース１０４ａの構文情報（雛形）３０３からＨＤＬ構文を生成する。このため設計者はＨＤＬの予約語だけでＨＤＬを記述することができ、ＨＤＬの文法を学習する負担を低減できシンタックスミスを防ぐことができる。

【００３５】入力画面５０１は設計者者がVHDLで回路のインターフェイス（入出力端子）を規定する予約語'entity'と入力したところである。そして設計者は構文生成指示（例えばキーボードのエスケープキーを押す）を行う。ＨＤＬ構文生成部１００ｃは文字列'entity'を受け取る。受け取った文字列'entity'を鍵にしてHDL構文データベース１０４ａの言語種別３０１'ＶＨＤＬ'、予約語３０２'entity'の検索を行う。検索の結果'entity'が検出されたら構文情報（雛形）３０３から、'entity'の雛形を生成する。入力画面５０２は'entity'の雛形が表示されたところを示す。

【００３６】設計者は生成された'entity'の雛形構文に回路設計で必要な情報（図５の'？'の部分）を入力し構文を完成させる。ＨＤＬ構文生成部１００ｃよりＨＤＬの雛形が生成され設計者者はシンタックスに気を使わずにすみ経験が少ない設計者でもシンタックスエラーを防ぐ事ができる。

【００３７】＜図６の説明＞HDL構文生成部１００ｃの処理フローを用いてＨＤＬ構文生成部１００ｃの処理を詳細に説明する。

【００３８】Ｓｔｅｐ６００は入力画面から入力された文字列が、HDLの予約語である文字列を鍵にHDL構文データベース１０４ａの言語種別３０１・予約語３０２を検索する。入力画面５０１の場合、言語種別３０１'VHDL'・予約語３０２'entity'を鍵にしてＨＤＬ構文データベース１０４ａの検索を行う。

【００３９】Ｓｔｅｐ６０１はHDL構文データベース１０４ａの検索結果の判定を行う。検索の結果該当する予約語３０２がHDL構文データベース１０４ａから検出されたならＳｔｅｐ６０２でHDL構文データベース１０４ａの構文情報（雛形）３０３から文字列（予約語）の雛形を入力画面上に表示する。入力画面５０２は'entity'の構文情報（雛形）３０３から'entity'の雛形を表示したことを示す。

【００４０】またＳｔｅｐ６０１でHDL構文データベース１０４ａの検索結果該当する予約語３０２が検出されなかった場合には、Ｓｔｅｐ６０３で該当する予約語（入力文字）が無いと言った趣旨の警告メッセージを出力する。

【００４１】Ｓｔｅｐ６０４は入力画面の入力モードに戻りその後設計者は生成したHDL構文の雛形に対して回路情報（図５の'？'の部分）の入力を行い記述を完成させる。

【００４２】＜図７の説明＞図７は、関連記述生成部１００ｄの原理をあらわす。

【００４３】ＨＤＬで階層化記述の場合に上位階層の回路記述に、下位階層の回路端子情報をもとに端子情報の記述を行う必要がある。ＨＤＬの経験がある設計者でも端子情報の記述もれや記述ミスをする可能性がある。関連記述生成部１００ｄは下位階層のＨＤＬテキストファイル１０４ｃが作成済みにより関連設計データベース１０４ｂに端子情報が既に登録済みならば、関連設計データベース１０４ｂの端子情報（端子名称４０２・端子方向４０３・端子属性４０４）とＨＤＬ構文データベース１０４ａからＨＤＬ構文を生成する。これにより端子の記述漏れや記述ミスを防ぐことができる。

【００４４】７０１は設計者が入力画面からVHDLで階層化を表現するの予約語'component'と入力し、その後に下位階層の回路名称（この例の場合ADD）を記述し構文生成指示（以下構文生成指示はキーボードのエスケープキーを押すことを意味する）を行う。次に関連設計データベース１０４ｂに回路'ＡＤＤ'が作成済みか回路名称４０１の検索を行う。検索の結果、下位階層記述'ＡＤＤ'が登録されていれば、関連設計データベース１０４ｂ回路名称４０１'ＡＤＤ'の端子名称４０２・端子方向４０３・端子属性４０４から'component'文に必要な端子情報を作成する。

【００４５】またＨＤＬ構文データベース１０４ａから予約語‘component’文の構文情報（雛形）３０３を検出し雛形をする。関連設計データベース１０４ｂから生成した端子情報とＨＤＬ構文データベース１０４ａから検出した構文情報（雛形）３０３をマージして７０２の入力画面に示す‘component’文を生成する。

【００４６】関連設計データベース１０４ｂに‘ＡＤＤ’回路情報がない場合には入力画面７０３の様に‘component’文の雛形のみの生成を行い‘？’の部分に設計者自身で端子情報の記述を行う。

【００４７】既に関連設計データベース１０４ｂに下位階層の回路情報が登録されている場合関連記述生成部１００ｄによりＨＤＬ構文が作成され記述洩れや記述ミスを防ぐことができる。

【００４８】＜図８の説明＞関連記述生成部１００ｄの処理フロー図を用いて関連記述生成部１００ｄの処理を詳細に説明する。

【００４９】Ｓｔｅｐ８００は関連設計データベース１０４ｂの回路名称４０１を検索する。入力画面７０１の場合は文字列‘ADD’を鍵に関連設計データベース１０４ｂの回路名称４０１に‘ADD’が登録されているか検索する。

【００５０】Ｓｔｅｐ８０１は検索した結果、回路名称４０１にデータが登録済みであれば、Ｓｔｅｐ８０２は関連設計データベース１０４ｂの端子情報（端子名称４０２・端子方向４０３・端子属性４０４）から‘component’構文のport文に必要なデータを作成する。入力画面７０２の場合は関連設計データベース１０４ｂの回路名称４０１‘ＡＤＤ’の端子情報から‘ＩＮ１‘，’ＩＮ２‘、’Ｓ‘の端子情報を抽出する。（図４の端子情報は’ＩＮ１‘しか記載していないが‘ＩＮ２’と‘Ｓ’の端子情報が‘ＩＮ１’の後ろに繰り返しある）Ｓｔｅｐ８０３はHDL構文データベース１０４ａから予約語‘component’文の構文情報（雛形）３０３を抽出し、Ｓｔｅｐ８０２で作成した端子情報とマージして‘component’文を完成させて入力画面に表示する。入力画面７０２は‘component’構文の雛形と端子情報（ＩＮ１，ＩＮ２，Ｓ）をマージして生成された‘component’文を示す。

【００５１】Ｓｔｅｐ８０１で関連設計データベース１０４ｂに関連記述が無い場合には、Ｓｔｅｐ８０４で該当する関連情報が無いと言った趣旨の警告メッセージを出力する。

【００５２】Ｓｔｅｐ８０５はHDL構文の雛形を生成するため、HDL構文生成部１００ｃに処理が移り、図６で説明した処理と同じように‘component’文の雛形を生成し入力画面に表示する。

【００５３】＜図９の説明＞図９は、関連記述修正部１００ｅの原理をあらわす。

【００５４】論理変更などにより回路の端子記述を変更する場合がある。その記述変更した回路を端子情報を他のＨＤＬテキストファイル１０４ｃで使用している場合変更した端子情報を記述している他のＨＤＬテキストファイル１０４ｃの端子記述も同じように変更をしなければならない。この様に変更個所の端子情報を他のＨＤＬテキストファイル１０４ｃで使用している場合、他のＨＤＬテキストファイル１０４ｃの変更漏れや変更ミスをする恐れがある。

【００５５】関連記述修正部１００ｅは関連設計データベス１０４ｂから修正個所が他のＨＤＬテキストファイル１０４ｃで使用していないか検索し使用していたら端子記述の変更を行う。この関連記述修正部１００ｅにより端子の記述修正により影響がある他のＨＤＬテキストファイル１０４ｃの端子の記述修正漏れや修正ミスを防ぐことができる。

【００５６】９０１は回路名称‘ADD’に出力端子‘OUT’を追加修正を示した図である。９０１ａは修正前の記述である。回路の端子は‘IN1’‘IN2’‘S’の３端子である。９０１ｂは９０１ａの記述に出力端子‘OUT’を追加修正し４端子に変更した事を示した図である。９０２は９０１の記述変更により回路‘ＡＤＤ’を参照している他のHDLテキストファイル１０４ｃである回路名称‘TOP’の出力端子に‘OUT’追加したことを示す図である。９０２は回路の階層を表現する記述である。９０２ａは上位階層記述で‘component’文により９０１ａの下位階層回路‘ADD’を用いた構造記述であり‘ＡＤＤ’の端子情報を共有している。９０２ｂは９０１ｂで示す様に出力端子‘OUT’を追加したことで‘component’文‘ADD’の‘port’文に出力端子‘OUT’を追加修正した事を示す。

【００５７】９０１ｂで端子の記述を変更したので関連設計データベース１０４ｂの端子情報（端子名称４０２・端子方向４０３・端子属性４０４）を更新する。９０２の場合関連設計データベース１０４ｂの回路名称４０１‘ADD’レコードに端子情報‘OUT’を追加する。

【００５８】次に関連設計データベース１０４ｂの回路名称４０１‘ADD’の関連回路情報４０５に回路名称があれば、関連回路情報４０５に登録されている回路名称のHDLテキストファイル１０４ｃをＨＤＬ入力プログラム１００ａに読み込み関連する端子記述の変更を行う。

【００５９】図９の場合、回路‘TOP’HDLテキストファイル１０４ｃの作成時に関連回路情報４０５に‘TOP’が登録される。回路‘TOP’のＨＤＬテキストファイル１０４ｃを読み込み出力端子‘OUT’の追加を行う。

【００６０】関連記述修正部１００ｅは既作成済みのＨＤＬテキストファイル１０４ｃの変更により他のＨＤＬテキストファイル１０４ｃの修正が必要か関連設計データベース１０４ｂから検索し、修正が必要なＨＤＬテキストファイル１０４ｃがあれば修正を行う。これにより変更もれや変更ミスを防ぐことができる。

【００６１】＜図１０の説明＞関連記述修正部１００ｅの処理フロー図を用いて関連記述修正部１００ｅの処理を詳細に説明する。

【００６２】Ｓｔｅｐ１０００は設計変更等により回路の入出力端子記述を変更したので、変更中の回路名称を鍵に関連設計データベース１０４ｂの回路名称４０１を検索する。図９の場合は回路名称４０１‘ADD’を鍵に関連設計データベース１０４ｂの回路名称４０１を検索する。

【００６３】Ｓｔｅｐ１０１０は関連設計データベース１０４ｂの回路名称４０１‘ＡＤＤ’レコードに端子情報‘OUT’の追加処理を行いデータを更新する。図９の場合、端子名称４０２‘OUT’・端子方向４０３‘out’・端子属性４０４‘std_logic’が追加される。

【００６４】Ｓｔｅｐ１０２０は関連設計データベース１０４ｂの関連回路情報４０５を検索し関連路情報４０５に回路名称が登録されているか検索する。図９の場合は関連回路情報４０５に回路名‘TOP’が登録されている。（関連回路情報４０５の登録は回路名称‘TOP’のHDLテキストファイル１０４ｃの作成時に‘component’文の‘ADD’を記述する際に関連記述生成部１００ｄで登録される）関連回路情報４０５に回路名称が登録されていればその回路名称のＨＤＬテキストファイル１０４ｃをＨＤＬ入力プログラム１００ａに読み込み、Ｓｔｅｐ１０１０で更新された端子情報（端子名称４０２・端子方向４０３・端子属性４０４）から変更データを作成して端子記述の変更を行う。図９の９０２ｂの場合は、端子情報‘OUT’が追加される。

【００６５】Ｓｔｅｐ１０４０はＳｔｅｐ１０２０の関連設計データベース１０４ｂの検索の結果、関連回路情報４０５に回路名称が登録されていないない場合は通常の入力モードに戻り設計者がＨＤＬ入力プログラム１００ａの入力画面から入力を行う。

【００６６】＜図１１の説明＞図１１は、セマンティックスチェック部１００ｆの原理をあらわす。

【００６７】回路設計では設計した回路が正しく動作するかシミュレータを用いてシミュレーションを行う。シミュレーションではVHDLの‘process’文、Ｖｅｒｉｏｌｇ HDLの‘always’文に含まれるセンシティビティリストの信号の変化を検証する。そのため信号代入文の右辺の信号、または条件式内の信号がセンシティビティリストに記述する必要がある。（ただしVHDLの場合‘wait’文の記述が‘process’文内にある場合は記述不可）ＨＤＬ記述の初心者でなくてもセンシティビティリストの記述もれをする恐れがあり回路の検証工程で問題となる。セマンティックスチェック部１００ｆはセンシティビティリストに必要な信号を検出してセンシティビティリストを作成する。これにより不完全なセンシティビティリストの作り込みを防ぎ回路の検証工程での問題を防ぐことが出来る。

【００６８】１１０１様にセンシティビティリストの記述にもれがある場合、１１０２のように信号代入文の右辺または条件式内の信号はセンシティビティリストに追加をする。（センシティビティリストとは‘process’文または、‘always’文の後ろの‘（）’内の事である）図１１の場合１１０１には‘B’‘SEL’がセンシティビティリストに記述されていないので、１１０２で示す様に‘B’‘SEL’をセンシティビティリストに追加する。

【００６９】図１１の１１０３は未使用な信号・端子のチェックを行う。１１０３では‘signal’宣言で‘A__ＳＩＧ’と記述されているが機能記述部（process文内）で使用されていない。このような記述を残しておくと不要な端子を生成し回路の面積が大きくなったりして回路の品質が悪くなる恐れがある。この様な未使用な信号・端子が存在した場合警告メッセージを出力する。設計者は警告メッセージにより記述に不要な端子や信号がないか確認がとれ回路の品質を低下することを防ぐことができる。

【００７０】＜図１２の説明＞セマンティックスチェック部１００ｆの処理フローを用いてセマンティックスチェック部１００ｆの処理を詳細に説明する。

【００７１】Ｓｔｅｐ１２００はHDL構文生成部１００ｃの処理により‘process’文や‘always’文の生成を行う。

【００７２】Ｓｔｅｐ１２１０は生成された‘process’や‘always’文にHDL入力プログラム１００ａの入力画面から設計者が回路の機能記述を行う。この時センシティビティリスト内の信号情報をワークエリア１００ｇにテーブルＡ、条件式や信号代入文の右辺の信号情報はワークエリア１００ｇにテーブルＢとして記憶しておく。

【００７３】Ｓｔｅｐ１２２０は機能記述が終了したら、センシティビティリスト内の信号情報（テーブルA）と、条件式号や信号代入文の右辺の信号情報（テーブルB）を比較してセンシティビティリストに記述もれがあればテーブルBからデータを作成してセンシティビティリストに追加する。

【００７４】図１１の１１０１の場合ワークエリア１００ｇのテーブルＡには‘A’が、ワークエリア１００ｇのテーブルＢには‘A’‘B’‘SEL’を記憶する。テーブルＡとテーブルＢの比較の結果テーブルＡに記憶れれていない‘B’‘SEL’をセンシティビティリストに追加する。

【００７５】Ｓｔｅｐ１２３０は信号宣言または入出力端子情報を信号・端子情報データとしてワークエリア１００ｇのテーブルCに記憶する。Ｓｔｅｐ１２４０は機能記述部の信号情報を信号情報データとしてワークエリア１００ｇのテーブルDに記憶する。

【００７６】Ｓｔｅｐ１２５０は、信号・端子情報デー

タ（テーブルC）と信号情報データ（テーブルD）を比較する。Ｓｔｅｐ１２５０はテーブルＣとテーブルＤを比較した結果登録データに過不足があった場合、過不足となった信号・端子情報について警告メッセージの出力を行う。

【００７７】図１１の１１０３の場合信号・端子情報データ（テーブルC）には‘port’情報‘I1’‘I2’‘SEL’‘O1’と、signal情報‘A_SIG’が記憶される。信号情報データ（テーブルD）には機能記述部（process文内）の‘I1’‘I2’‘SEL’‘O1’が記憶される。テーブルＣとテーブルＤの比較結果‘A_SIG’が信号宣言しているが未使用な信号と判断され‘A_SIG’に対して警告メッセージを出力する。

【００７８】

【発明の効果】以上述べた如く本発明を用いれば、HDLを用いて論理設計を行う上で、前記従来手法の問題点を解決し、HDL記述を用いて設計を行う事ができる。

【図面の簡単な説明】

【図１】HDL入力システムのシステム構成図である。

【図２】HDL入力システムのプログラム構成図である。

【図３】HDL構文データベースのデータ構造をあらわす。

【図４】関連設計データベースのデータ構造をあらわす。

【図５】HDL構文生成部の原理をあらわす。

【図６】HDL構文生成部の処理フロー図である。

【図７】関連記述生成部の原理をあらわす。

【図８】関連記述生成部の処理フロー図である。

【図９】関連記述修正部の原理をあらわす。

【図１０】関連記述修正部の処理フロー図である。

【図１１】セマンテックチェック部の原理をあらわす。

【図１２】セマンテックチェック部の処理フロー図である。

【符号の説明】

１００…主記憶装置、
１００ａ…HDL入力システム、
１００ｂ…HDL入出力部、
１００ｃ…HDL構文生成部、
１００ｄ…関連記述生成部、
１００ｅ…関連記述修正部、
１００ｆ…セマンティックスチェック部、
１０１…ＣＰＵ、
１０２…ディスプレイ、
１０３…キーボード、
１０４…ハードディスク、
１０４ａ…ＨＤＬ構文データベース、
１０４ｂ…関連設計データベース、
１０４ｃ…HDLテキストファイル、
１０５…バス、
３０１…言語種別、
３０２…予約語、
３０３…構文情報（雛形）、
４０１…回路名称、
４０２…端子名称、
４０３…端子方向、
４０４…端子属性、
４０５…関連回路情報、
５０１…初期入力画面、
５０２…ＨＤＬ構文生成画面、
７０１…初期入力画面、
７０２…自動生成画面（関連情報あり）、
７０３…自動生成画面（関連情報なし）、
９０１…記述変更側入力画面、
９０１ａ…記述変更前入力画面、
９０１ｂ…記述変更後入力画面、
９０２…関連記述側入力画面、
９０２ａ…自動変更前入力画面、
９０２ｂ…自動変更後入力画面、
１１０１…自動生成前入力画面、
１１０２…自動生成後入力画面、
１１０３…セマンティックチェックスの記述例。

【図３】

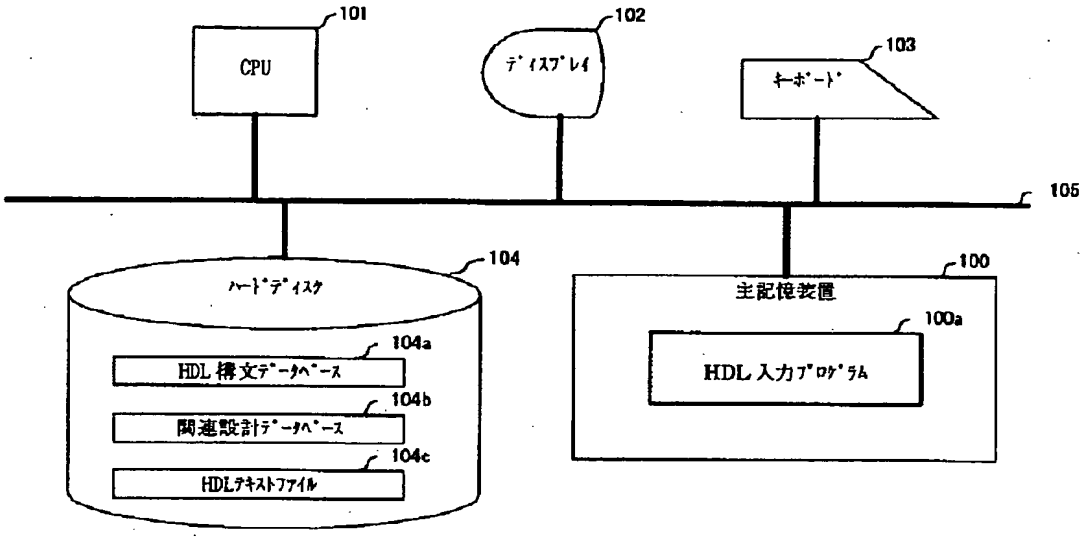
図３

104a

301 言語種別	302 予約語	303 構文情報（雛形）
VHDL	entity	entity ??? is generic(???:??? in ???:=???); port(???:in ???; ??? out ???); end ???;
Verilog	module	module ??? (???,???); endmodule
:	:	:

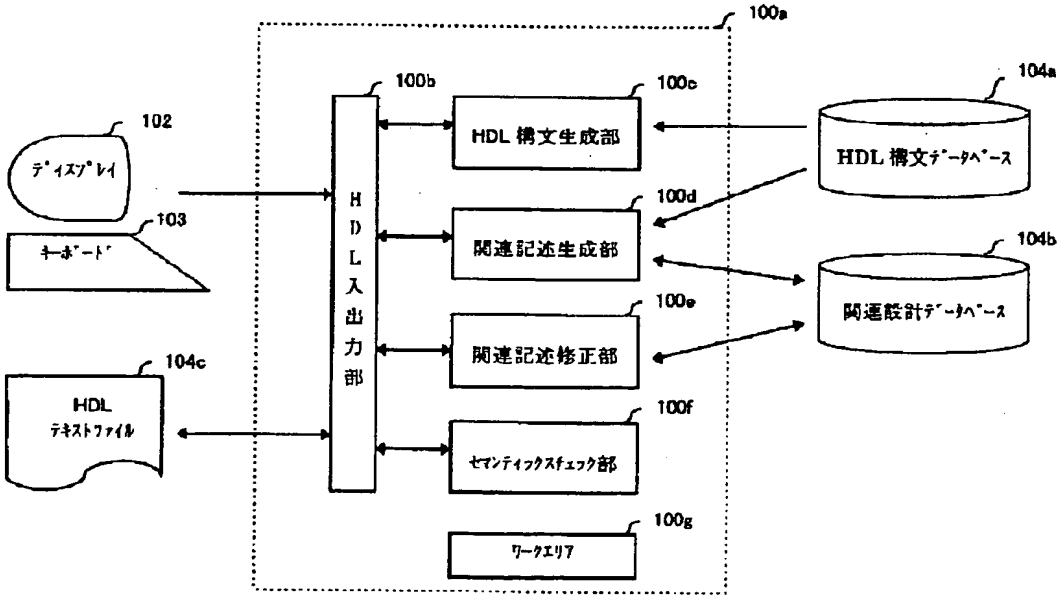
【図１】

図１


101
CPU
102
ﾃﾞｨｽﾌﾟﾚｲ
103
ｷｰﾎﾞｰﾄﾞ
105
104
ﾊｰﾄﾞﾃﾞｨｽｸ
104a
HDL構文ﾃﾞｰﾀﾍﾞｰｽ
104b
関連設計ﾃﾞｰﾀﾍﾞｰｽ
104c
HDLﾃｷｽﾄﾌｧｲﾙ
100
主記憶装置
100a
HDL入力ﾌﾟﾛｸﾞﾗﾑ

【図２】

図２


100a
102
ﾃﾞｨｽﾌﾟﾚｲ
103
ｷｰﾎﾞｰﾄﾞ
104c
HDL
ﾃｷｽﾄﾌｧｲﾙ
100b
HDL入出力部
100c
HDL構文生成部
100d
関連記述生成部
100e
関連記述修正部
100f
ｾﾏﾝﾃｨｯｸｽﾁｪｯｸ部
100g
ﾜｰｸｴﾘｱ
104a
HDL構文ﾃﾞｰﾀﾍﾞｰｽ
104b
関連設計ﾃﾞｰﾀﾍﾞｰｽ

【図４】

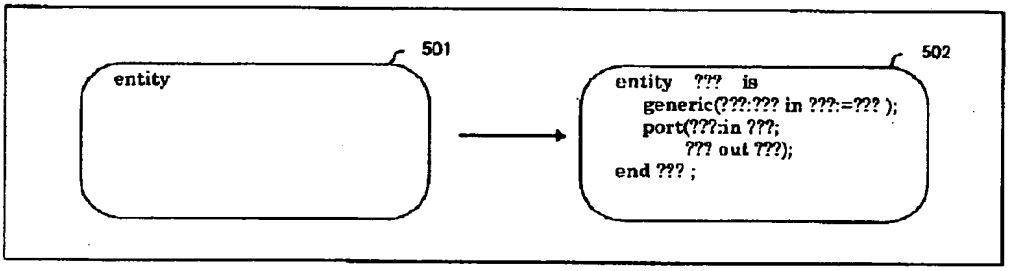
図４

401 回路名称	402 端子名称	403 端子方向	404 端子属性	・・・	405 関連回路情報 (104b)
ADD	IN1	in	std_logic		top
:	:	:	:	:	:

端子数分 402・403・404 の繰り返

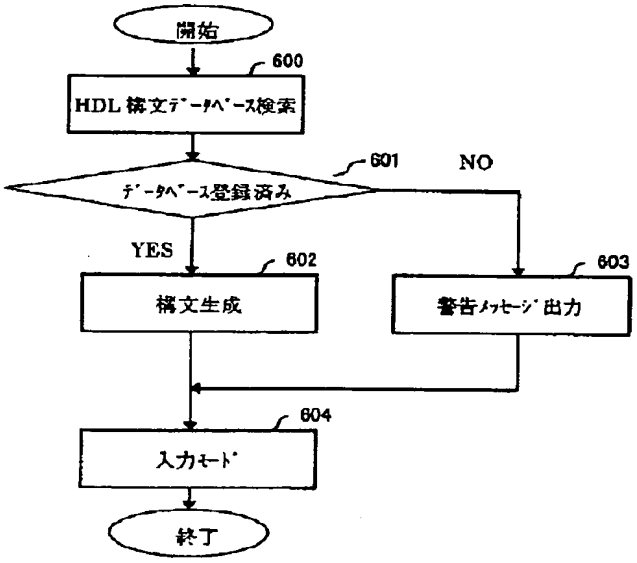
【図５】

図５



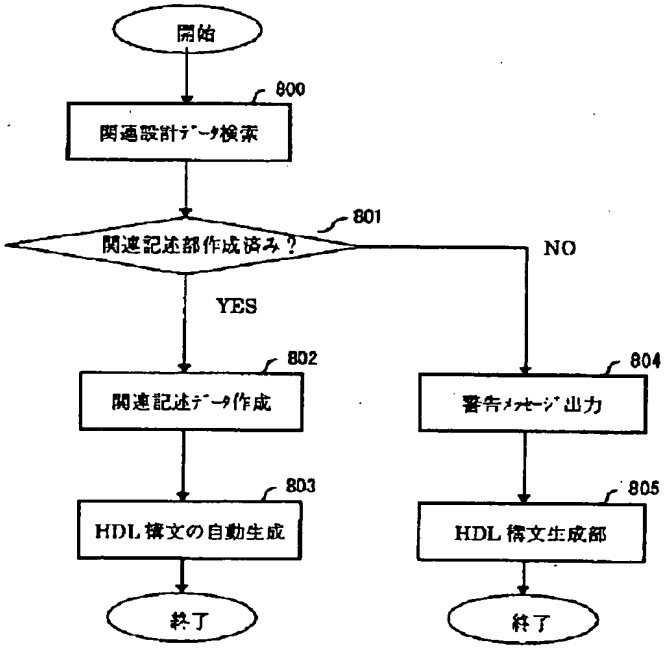
【図６】

図６



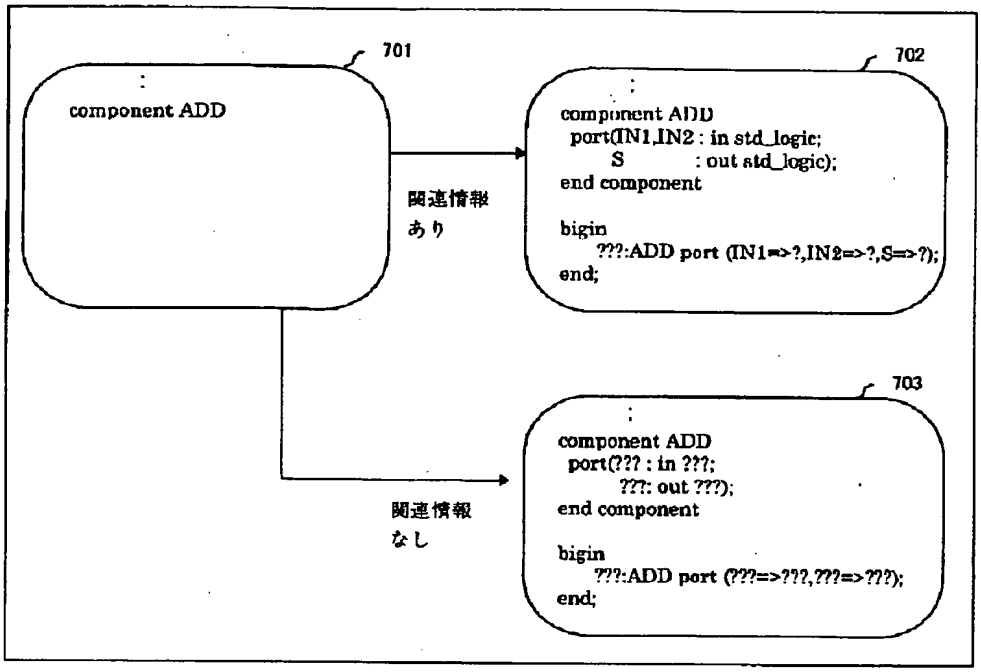
【図８】

図８



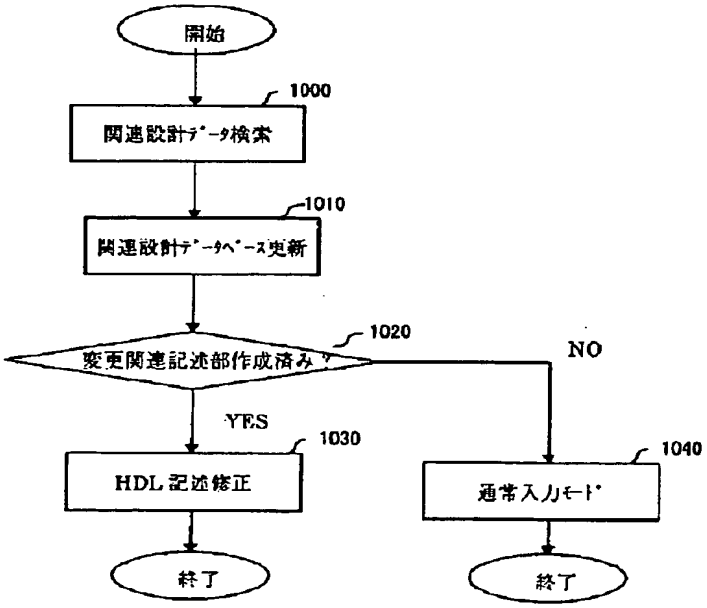
【図７】

図７


701
component ADD
関連情報
あり
702
component ADD
port(IN1,IN2 : in std_logic;
S : out std_logic);
end component
bigin
???:ADD port (IN1=>?,IN2=>?,S=>?);
end;
関連情報
なし
703
component ADD
port(??? : in ???;
???: out ???);
end component
bigin
???:ADD port (???=>???,???=>???);
end;

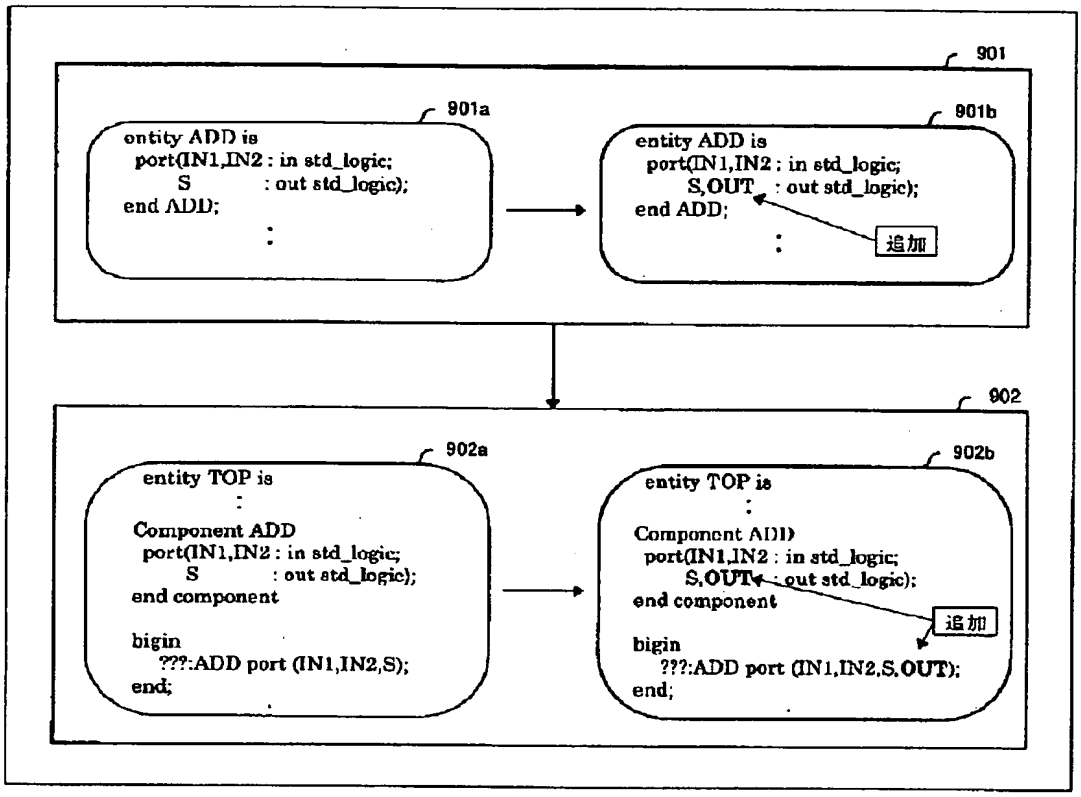
【図１０】

図 10


開始
1000
関連設計ﾃﾞｰﾀ検索
1010
関連設計ﾃﾞｰﾀﾍﾞｰｽ更新
1020
変更関連記述部作成済み？
NO
YES
1030
HDL記述修正
1040
通常入力ﾓｰﾄﾞ
終了
終了

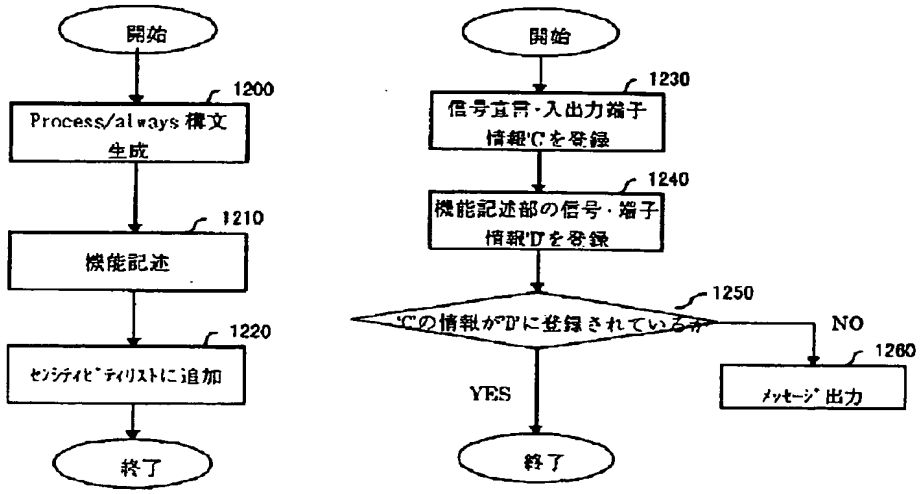
【図９】

図９


901
901a
entity ADD is
port(IN1,IN2 : in std_logic;
S : out std_logic);
end ADD;
:
901b
entity ADD is
port(IN1,IN2 : in std_logic;
S,OUT : out std_logic);
end ADD;
:
追加
902
902a
entity TOP is
:
Component ADD
port(IN1,IN2 : in std_logic;
S : out std_logic);
end component
bigin
???:ADD port (IN1,IN2,S);
end;
902b
entity TOP is
:
Component ADD
port(IN1,IN2 : in std_logic;
S,OUT : out std_logic);
end component
追加
bigin
???:ADD port (IN1,IN2,S,OUT);
end;

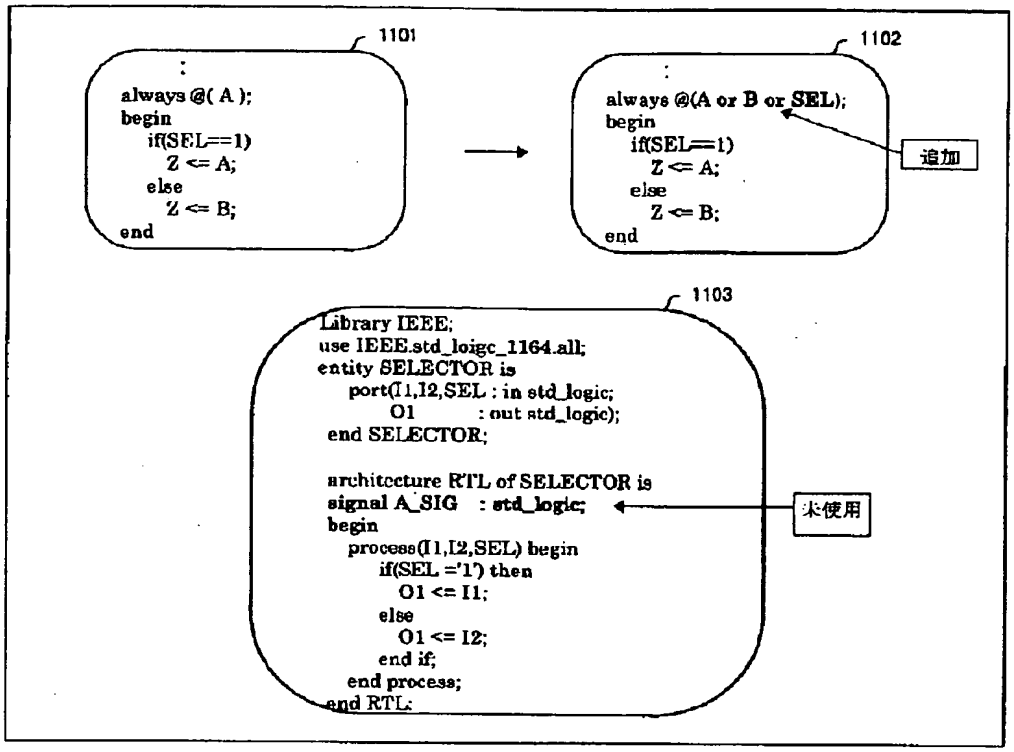
【図１２】

図 12


開始
1200
Process/always 構文
生成
1210
機能記述
1220
ｾﾝｼﾃｨﾋﾞﾃｨﾘｽﾄに追加
終了
開始
1230
信号宣言・入出力端子
情報'C'を登録
1240
機能記述部の信号・端子
情報'D'を登録
1250
'C'の情報が'D'に登録されているか
NO
YES
1260
ﾒｯｾｰｼﾞ出力
終了

【図１１】

図 11


1101
:
always @(A);
begin
if(SEL==1)
Z <= A;
else
Z <= B;
end
1102
:
always @(A or B or SEL);
begin
if(SEL==1)
Z <= A;
else
Z <= B;
end
追加
1103
Library IEEE;
use IEEE.std_loigc_1164.all;
entity SELECTOR is
port(I1,I2,SEL : in std_logic;
O1 : out std_logic);
end SELECTOR;
architecture RTL of SELECTOR is
signal A_SIG : std_logic;
begin
process(I1,I2,SEL) begin
if(SEL ='1') then
O1 <= I1;
else
O1 <= I2;
end if;
end process;
end RTL;
未使用